Spécifications

Weather

- Processus fils de **Market**.
- A chaque fois qu'il se réveille, **weather** calcule une température **T** selon la formule suivante :

$$T' = \cos(w) * T * \alpha$$

$$w = w + 0.1 \qquad \alpha = random(-4\,;4)$$

- Et il calcule un entier aléatoire **E** entre 1 et 3 :
  - 1 – Soleil
  - 2 – Nuage, Pluie
  - 3 – Neige si **T** < 0°C
- Modifie le tableau de la **mémoire partagée** avec **Market** :
  - Meteo[0] = T
  - Meteo[1] = E

Bonus

- Recherche d'une meilleure formule pour la température et le temps, gestion des incohérences, gestion des saisons.

Market

- Main processus
- Processus multithreadé
- Père de **external**, **weather**, et les multiples **home**
- A chaque fin de journée, **market** calcule le prix de l'énergie pour la journée :

$$P_t = \gamma P_{t-1} + \sum_i (\alpha_i f_{i,t}) + \sum_j (\beta_j u_{(j,t)})$$

α, β, γ sont des constantes à déterminer.

Les f(i) sont stockées dans une liste **day_transactions**.

- **Market** écoute un objet de type **messageQueue** qui représente les demandes d'achat et de vente des maisons. Les objets stockés dans cette queue sont des entiers signés avec comme convention :
    - « + » = je veux vendre
    - « - » = je veux acheter

  Pour chaque case de cette queue :
    - (Si il reste des « places » de transaction possible, crée un nouveau thread « **worker** » pour gérer la transaction.)
    - Le thread envoie la quantité d'énergie achetée ou vendue dans une liste **day_transactions.**
    - Le **worker** se fini et efface sa mémoire.
- Gère les signaux entrants venant de **external** : alimente une liste de coefficients entiers à prendre en compte dans le calcul du prix.

Bonus

- ???

Home

- Plusieurs instances de ce processus possibles.
- Au lancement du thread, détermine des constantes aléatoires :
    - **Px** : taux de production d'énergie [[1, 20]]
    - **Cx** : taux de consommation d'énergie [[1,20]]
    - Politique énergétique {1,3}
        - 1 → Donne toujours son surplus d'énergie aux voisins
        - 2 → Vend toujours son énergie au **market**
        - 3 → Essaie de donner aux voisins pendant 1 jours, si pas de preneur, vend au **market.**
- A chaque fois qu'il se réveille :
    - Calcule son bilan énergétique : Q = Px – Cx
    - Si Q > 0, décide d'une action en fonction de sa politique
    - Si Q = 0, Ne fait rien
    - Si Q < 0,

    - Regarde si un voisin peut donner de l’énergie
    - Si non, achète une quantité Q d’énergie sur le marché
- Chaque **home** accède à une file de message d’énergie « à donner ». Elle peuvent la lire et l’écrire.
- **home** alimente une communication de type **messageQueue** avec **market**, typé en entiers signés selon la même convention de signe.

Bonus

- Ajouter des Pi et Ci variables (en fonction de la météo par exemple (ceci est possible car home est l’enfant de **market** et a donc accès à sa mémoire partagée))
- Calculer les Px et Cx à la création en fonction d’équipements réels (lave-linge, panneau solaire, serveurs google…)

External

- Processus fils de **market**
- Génère en interne un événement aléatoirement parmi des événements programmés.
- Si l’événement n’est pas nul, envoie au processus **market** un **signal** qui comporte un int symbolisant la gravité de la crise.